

Colorbit SDK

Colorbit SDK Programming Manual

2012/05/14

Version 1.1.3

**The content of this document is highly confidential
and should be handled accordingly.**

Confidential

These coded instructions, statements, and computer programs contain proprietary information of Nintendo and/or its licensed developers and are protected by national and international copyright laws. They may not be disclosed to third parties or copied or duplicated in any form, in whole or in part, without the prior written consent of Nintendo.

Table of Contents

1	About This Document	4
2	Overview	5
2.1	Module Description	5
2.1.1	Namespace mw::cb	5
2.1.2	Namespace mw::cb::Decoder	5
2.2	Build Environment Configuration	5
2.2.1	Add the Include Path	5
2.2.2	Add the Library File	5
3	Colorbit SDK Usage Guide	6
3.1	Include File	6
3.2	Initialization and Finalization	6
3.3	Parameter Settings	6
3.3.1	Decoder Parameters	7
3.3.2	Color Extraction Position Configuration	7
3.4	Colorbit Recognition	8
4	Sample Demo	10
4.1	Simple	10
4.2	cbt	10
	Revision History	11

Code

Code 2-1	Add the Include Path	5
Code 2-2	Add the Library File	5
Code 3-1	Include cbDecoder.h in the Source Code	6
Code 3-2	Example of Colorbit Initialization	6
Code 3-3	Example of Colorbit Finalization	6
Code 3-4	Example of Acquiring the Work Size	6
Code 3-5	Example of Decoder Parameter Settings	7
Code 3-6	Example of Color Extraction Position Configuration	7
Code 3-7	Example of Colorbit Recognition	8

Figures

Figure 3-1	Example of Color Extraction Position Configuration	8
Figure 3-3	Color Separation Results Data Format	9
Figure 4-1	Input Data Created from the NAR Library	10

1 About This Document

These programming guidelines explain basic programming using the Colorbit SDK for CTR API.

This document assumes the reader is familiar with C/C++ programming.

For details on the Colorbit SDK API, see the Function Reference.

For an overview on Colorbit, see the *Colorbit SDK Development Guide for Applications That Use Colorbit*.

2 Overview

The Colorbit library is an SDK prepared for ID recognition using Colorbit. By using this library, the Colorbit ID (roughly 1050) can be acquired from the image data.

See Chapter 3 Creating Colorbit in the *Colorbit SDK Development Guide for Applications That Use Colorbit* regarding Colorbit creation.

This section describes the modules for using the Colorbit library and the information required to build applications.

2.1 Module Description

2.1.1 Namespace mw::cb

This namespace defines only the error codes returned from the Colorbit API and the parameter values to be set.

2.1.2 Namespace mw::cb::Decoder

This namespace defines the function group in order to perform Colorbit recognition from the input image.

2.2 Build Environment Configuration

2.2.1 Add the Include Path

"\$(CTRMW_CBIT_ROOT)/include/mw/cb" must be added to the compiler include path.

Code 2-1 Add the Include Path

```
INCLUDES += $(CTRMW_CBIT_ROOT)/include/mw/cb
```

Note: The code example is content that should be included in `OMakefile` when using the CTR-SDK build system.

2.2.2 Add the Library File

The library file that should be passed to the linker is `libmw_cb_decoder.*.a` (* is either `fast` or `small`).

Code 2-2 Add the Library File

```
LIBFILES = $(addprefix  
$(CTRMW_CBIT_ROOT)$(DIRSEP)libraries$(DIRSEP)$(config.  
getTargetSubDirectoryture)$(DIRSEP), libmw_cb_decoder
```

Note: The code example is content that should be included in `OMakefile` when using the CTR-SDK build system.

3 Colorbit SDK Usage Guide

This chapter describes how to implement applications that use the Colorbit library referencing the process flow.

In addition, the API supplied by this library is not thread-safe.

3.1 Include File

The `cbDecoder.h` file must be included in order to use the Colorbit SDK features.

Code 3-1 Include `cbDecoder.h` in the Source Code

```
#include "cbDecoder.h"
```

3.2 Initialization and Finalization

When using the Colorbit recognition feature, the `Initialize` method must be called in advance. This method provides functions to allocate and deallocate work memory to be used internally by the library.

Code 3-2 Example of Colorbit Initialization

```
void* Malloc(size_t size);  
void* Free(void* ptr);  
  
mw::cb::Decoder::Initialize(Malloc, Free);
```

When exiting, call the `Finalize` method.

Code 3-3 Example of Colorbit Finalization

```
mw::cb::Decoder::Finalize();
```

In addition, specify the input image size in the `GetWorkSize` method to obtain the actual work memory size used.

Code 3-4 Example of Acquiring the Work Size

```
int image_width; //input image width  
int image_height; //input image height  
  
size_t worksize = mw::cb::Decoder::GetWorkSize(image_width, image_height);
```

3.3 Parameter Settings

Setting the parameters in advance allows using the Colorbit recognition feature to match the design to be used and the intended purpose.

3.3.1 Decoder Parameters

If the input image can be expected to be dark because of the photograph environment influence, the dark location correction can be set to ON/OFF with `mw::cb::IMAGE_CORRECTION`.

Note: The dark location correction process also works even when the input image is sufficiently bright. In this case, there is little impact on the recognition rate but the processing time increases. If the photograph environment can be fixed to a bright location in advance, we recommend setting the dark location correction process to OFF.

Furthermore, the error correction level used during Colorbit recognition can be set to three levels, 0 to 2, with `mw::cb::EC_LEVEL`.

- Error Correction Level 0: No error correction
- Error Correction Level 1: Correction of one color error
- Error Correction Level 2: Correction of two color errors

Code 3-5 Example of Decoder Parameter Settings

```
// dark location correction OFF, error correction level 2
mw::cb::Decoder::SetParams(mw::cb::DECODER_NONUSE_IMAGE_CORRECTION,
                           mw::cb::CODING_LD5_C15_CORRECTION_LEVEL_2);
```

3.3.2 Color Extraction Position Configuration

Colorbit is expressed with 15 color arrays.

By using the `SetColorLocation` method to specify these 15 color acquisition positions, the configuration can be made to match the design to be used.

The specified coordinate system uses the upper left vertex as the origin and specifies each extraction point with the horizontal direction as the first element and the vertical direction as the second element.

Specify the coordinate value as the absolute coordinate in the input image.

Also, with `mw::cb::Decoder::QueryImage`, the color array is created in the element order of `location_table`.

When capturing a 64 X 64 pixel image using the sample codes, the color extraction position configuration is as follows.

Code 3-6 Example of Color Extraction Position Configuration

```
int location_table[15][2] = {
    {17,15},{28,15},{38,15},{49,15},{60,15},
    {12,32},{22,31},{33,31},{44,31},{54,31},
    {6,49},{17,49},{28,49},{38,49},{49,49}
};

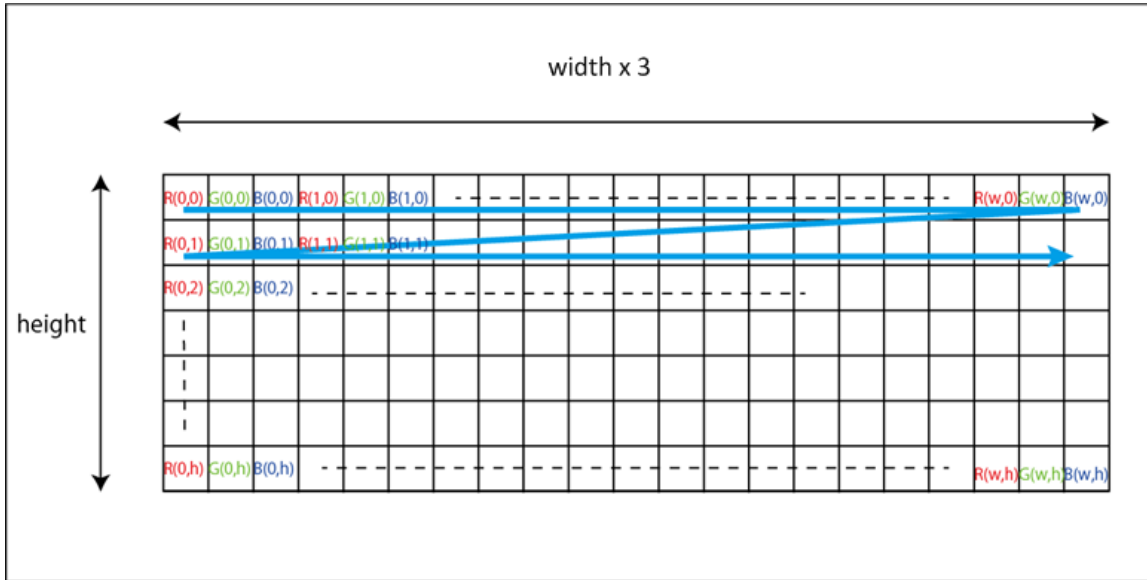
mw::cb::Decoder::SetColorLocation(location_table);
```

Figure 3-1 Example of Color Extraction Position Configuration

3.4 Colorbit Recognition

After setting the parameters, input the image data for which Colorbit has been captured into the `mw::cb::Decoder::QueryImage` method to perform ID recognition.

The input image data uses the pixel format RGB888 linear format.

Figure 3-2 Input Image Data Format

Color arrays obtained with the `mw::cb::Decoder::QueryImage` method are expressed as follows.

R = 0, G = 1, B = 2, NO_COLOR = 255

Code 3-7 Example of Colorbit Recognition

```
(void*) image_data; //Pixel Format = RGB888
int image_width;
int image_height;

// For Colorbit id acquisition, allocate an unsigned long long type
// (64-bit) data region
unsigned long longcb_id;
// For Colorbit color array acquisition, allocate unsigned char type
// (8-bit)*15 data regions
unsigned char color_seq[15];

int error = mw::cb::Decoder::QueryImage(image_data, width, height,
&cb_id, &color_seq);

if(error < 0){
    //Colorbit Decode Error
}else{
```



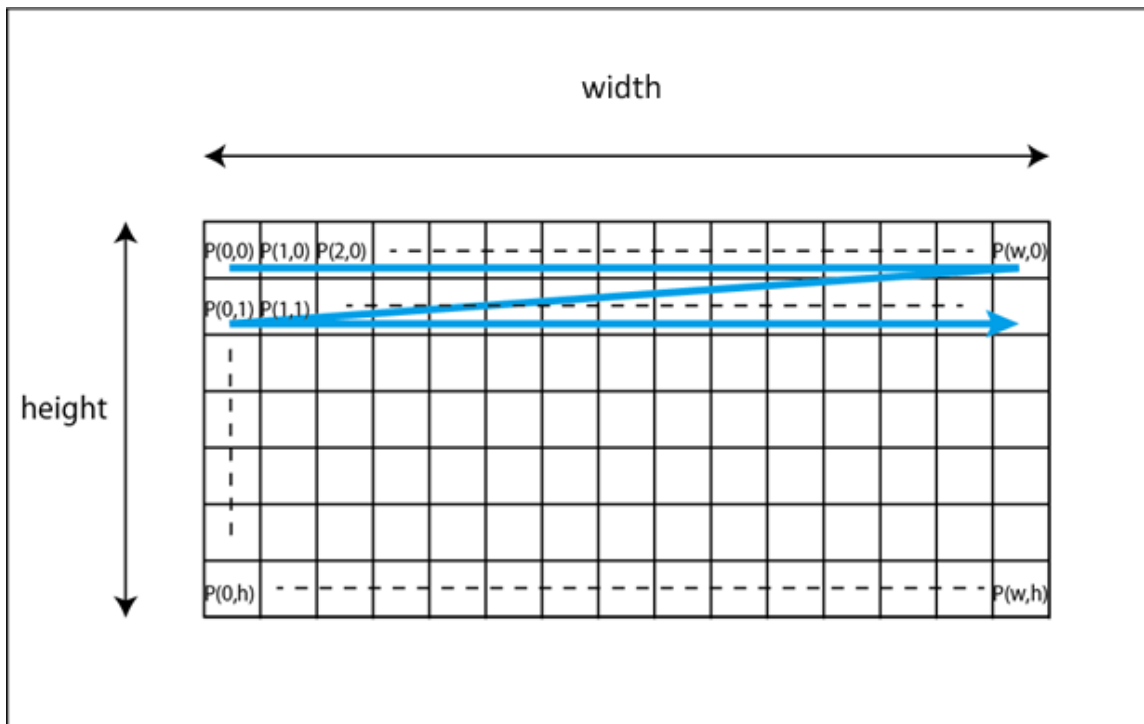
```
}

```

In addition, use the `mw::cb::Decoder::QueryImageEx` method to obtain the entire input image color separation results.

The color separation results are expressed as 8-bit image data of R = 150, G = 100, B = 50, NO_COLOR = 0.

Figure 3-3 Color Separation Results Data Format



Code 3-8 Example 2 of Colorbit Recognition

```
unsigned char* color_dump;
color_dump =
(unsigned char*)malloc(sizeof(unsignedchar)*image_width*image_height);

int error = mw::cb::Decoder::QueryImageEx(image_data, width, height,
&cb_id, &color_seq, color_dump);

free(color_dump);
```

4 Sample Demo

4.1 Simple

In the sample demo Simple, you get an image from the camera and capture the Colorbit with the correct orientation within the camera preview frame displayed on the upper screen and at the color extraction location to recognize the Colorbit.

4.2 cbit

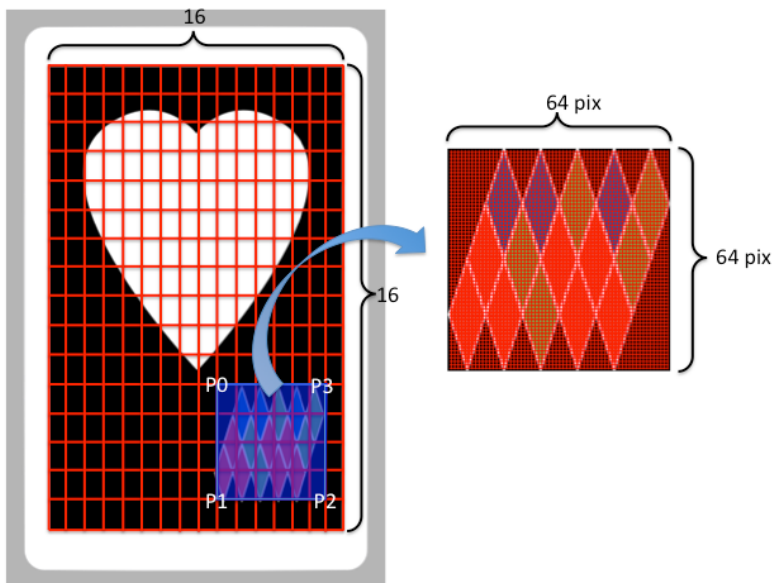
In the sample demo cbit, you use the NAR library to recognize Colorbit.

With this library, because there is no function that detects the Colorbit position and orientation from an image, Colorbit recognition is performed with marker detection that uses the NAR library and information from within the detected area.

This works as follows.

1. The NAR library performs marker detection and the acquired area is divided vertically and horizontally by 16.
2. Specify four appropriate points from the points created by the divisions.
3. Divide the area by 64 again and get the color data of the intersections.
4. Use the acquired 64 X 64 pixel data as the input data and run Colorbit recognition.

Figure 4-1 Input Data Created from the NAR Library



Revision History

Version	Revision Date	Category	Description
1.1.3	2012/05/14	Changed	Changed the version number.
1.1.2	2012/04/04	Added	4 Sample Demo
		Changed	3.3.2 Color Extraction Position Configuration Added sample code setting example.
1.0.1	2012/02/28	—	Changed the version number.
1.0.0	2011/12/22	—	Initial version.

Colorbit is the registered trademark of B.CORE Inc .

All other company and product names in this document are the trademarks or registered trademarks of their respective companies.

© 2012 Nintendo

The contents of this document cannot be duplicated, copied, reprinted, transferred, distributed, or loaned in whole or in part without the prior approval of Nintendo.